Efficient Parallel Resolution of The Simplified Transport Equations in Mixed-Dual Formulation

M. Barrault[a], B. Lathuilière[a], P. Ramet[b], J. Roman[c]

[a] *EDF R&D 1 Avenue du Général De Gaulle 92140 CLAMART, FRANCE*
[b] *INRIA Bordeaux Sud-Ouest Equipe-Projet Bacchus, FRANCE*
[c] *INRIA Bordeaux Sud-Ouest Equipe-Projet HiePACS, FRANCE*

Abstract

A reactivity computation consists of computing the highest eigenvalue of a generalized eigenvalue problem, for which an inverse power algorithm is commonly used. Very fine modelizations are difficult to treat for our sequential solver, based on the simplified transport equations, in terms of memory consumption and computational time.

A first implementation of a Lagrangian based domain decomposition method brings to a poor parallel efficiency because of an increase in the power iterations [1]. In order to obtain a high parallel efficiency, we improve the parallelization scheme by changing the location of the loop over the subdomains in the overall algorithm and by benefiting from the characteristics of the Raviart-Thomas finite element. The new parallel algorithm still allows us to locally adapt the numerical scheme (mesh, finite element order). However, it can be significantly optimized for the matching grid case. The good behavior of the new parallelization scheme is demonstrated for the matching grid case on several hundreds of nodes for computations based on a pin-by-pin discretization.

Key words: Simplified Transport Equation, Raviart-Thomas Finite Element, HPC, Parallelism, Domain Decomposition Method

1. Introduction

The operation of a PWR-based[1] nuclear plant requires its fuel to be changed every 18 months. This must be done while ensuring the safety and the productivity of the plant in service. More precisely, in our context, EDF uses the numerical simulation of the neutron transport inside a nuclear reactor. Hence, EDF has developed a fast sequential solver [2] based on the simplified transport equations [3, 4][2].

The mid-term goal is to run efficiently large scale simulations[3] based on the simplified transport equations. In this context, the sequential algorithm suffers of two limitations. On the one hand, we are not able to run efficiently large scale computations due to memory consumption and/or computational time. On the other hand, it is necessary to refine a large part of the mesh when a better numerical approximation is needed in a local part of the reactor core. So, to tackle these problems, in [5, 1] we adapted a non overlapping domain decomposition method based on Lagrange multipliers to the simplified transport equations. In order to get the most generic algorithm, an approach with multigroup solver was chosen. As this approach leads to a significant increase in the number of power iterations, a good scalability could not be obtained. In the present article, we obtain a very good parallel efficiency while improving the parallelization scheme of Lagrangian domain decomposition method applied to the sequential power algorithm made of four nested loops. First, we change the location of the loop over the subdomains to obtain the same number

[1] Pressurized Water Reactor.
[2] These equations are called SPn.
[3] Like pin-by-pin discretization of SP3/SP5 equations with 26 energy groups.

of power iterations as the sequential algorithm. Secondly, we benefit from the characteristics of the Raviart-Thomas finite element to optimize the implementation of the new parallel algorithm.

In section 2, the diffusion equation and the sequential algorithm based on the same algorithm as used by the Minos solver [6, 7], are presented. As the domain decomposition strategy proposed in this paper is performed at the level of one diffusion system, the presentation is performed on the diffusion equation instead of SPn equations. Then in section 3, after an overview of previous works on the parallelization of the sequential algorithm, the Lagrange multipliers domain decomposition method and our new parallel algorithm are introduced. In section 4, the characteristics of Raviart-Thomas finite elements are used in the case of matching grids to get an efficient parallel implementation. Finally, in section 5, numerical results on the IAEA-3D benchmark and on data from real core computations are provided and analyzed; compared to the previous approach described in [1], the performance improvement factor is between 5 and 20, depending on the test case.

2. Background on SPn equations

The neutron flux is the solution of the Boltzmann transport equation which expresses the neutron balance between streaming, scattering and absorption. The flux depends on seven variables (one for the time, one for the energy[4], two for the direction and three for the space position). For the time variable, the steady state case is considered; it leads to a reactivity computation which consists of computing the highest eigenvalue of a generalized eigenvalue problem.

Numerical approximations are required to solve the Boltzmann equation. The discretization of the energy is done with the multigroup theory [8]: the energy variable is decomposed on N_g energy intervals (called energy groups). For the angular variable, several approximations can be used. Two of them are

- the Pn approximation: the flux is expanded on a basis of spherical-harmonics functions up to the order n. The method requires to compute $(n+1)^2$ spatial scalar fields for each energy group;
- the diffusion approximation: the Fick's law introduces a diffusion coefficient that links the flux to his gradient. This method requires to compute 4 spatial scalar fields for each energy group.

As the Pn approximation is expensive in terms of computational time, the simplified transport equations (SPn) have been developed [3, 4]. To obtain these equations, the neutron flux is supposed locally plane and is expanded on the 1D spherical-harmonics basis. The SPn equations lead to $\frac{n+1}{2}$ coupled diffusion systems[5] for each energy group. For the SP1 case, which is equivalent to the P1[6] case, a single diffusion system is obtained for each energy group. As the domain decomposition strategy proposed in this paper is performed at the level of one diffusion system, the presentation is performed on the multigroup diffusion equation. The method can be generalized without any difficulties to SPn equations (in section 5, results with SP1 and SP3 equations are presented).

2.1. Multigroup diffusion equation

To study the existence of a solution of the steady state case, the following multigroup eigenvalue problem [8] has to be solved:

[4] The energy of a neutron is directly linked to its speed.

[5] $\frac{n+1}{2}$ is the number of harmonics.

[6] In the following, we prefer to use SP1 denomination to P1 because for $n \geq 3$ the domain decomposition method described in this paper applies to SPn equations but not for Pn equations.

Problem 1. *Find k_{eff} the highest positive eigenvalue such that it exists N_g strictly positive functions $\psi = (\phi_1, \ldots, \phi_{N_g})$ satisfying the following coupled system for each $1 \leq g \leq N_g$:*

$$-\text{div}\left(D_g \overrightarrow{\nabla} \phi_g\right) + \Sigma_{t,g}\phi_g - \sum_{g'} \left(\Sigma_s^{g' \to g} \phi_{g'}\right) = \frac{1}{k_{eff}} \left(\frac{\mathcal{X}_g}{4\pi}\right) . \sum_{g'} \left(\nu \Sigma_{f,g'} \phi_{g'}\right) \tag{1}$$

where:

- *k_{eff} is the effective multiplicative coefficient which characterizes the criticity of the nuclear core. $k_{eff} = 1$ means that there exists a solution to the steady state case. From a physical point of view, this means that the chain reaction is stable. $k_{eff} < 1$ (respectively $k_{eff} > 1$) means the problem is no longer steady state, as the number of neutrons is decreasing (respectively increasing);*
- *ϕ_g is the neutron flux in the group g;*
- *D_g is the diffusion coefficient in the group g;*
- *$\Sigma_{t,g}$ is the total cross section in the group g;*
- *$\Sigma_s^{g' \to g}$ is the scattering cross section from group g' to group g;*
- *$\mathcal{X}_g$ is the fission spectrum;*
- *$\nu\Sigma_{f,g}$ is the fission production in the group g.*

By denoting the fission operator by $\mathcal{F}$ and the transport operator by $\mathcal{H}$:

$$\mathcal{F} : \begin{pmatrix} \phi_1 \\ \vdots \\ \phi_{N_g} \end{pmatrix} \longrightarrow \left(\frac{1}{4\pi} . \sum_{g'} \nu \Sigma_{f,g'} \phi_{g'}\right) \begin{pmatrix} \mathcal{X}_1 \\ \vdots \\ \mathcal{X}_{N_g} \end{pmatrix}$$

$$\mathcal{H} : \begin{pmatrix} \phi_1 \\ \vdots \\ \phi_{N_g} \end{pmatrix} \longrightarrow \begin{pmatrix} -\text{div}\left(D_1 \overrightarrow{\nabla} \phi_1\right) + \Sigma_{t,1}\phi_1 - \sum_{g'} \Sigma_s^{g' \to 1} \phi_{g'} \\ \vdots \\ -\text{div}\left(D_{N_g} \overrightarrow{\nabla} \phi_{N_g}\right) + \Sigma_{t,N_g}\phi_{N_g} - \sum_{g'} \Sigma_s^{g' \to N_g} \phi_{g'} \end{pmatrix}$$

the N_g equations (1) can be written as:

$$\mathcal{H}\psi = \frac{1}{k_{eff}} \mathcal{F}\psi. \tag{2}$$

A Generalized Power Inverse Iteration algorithm [8] is used to solve (2), which implies solving $\mathcal{H}\psi^{p+1} = S^p$ at each power iteration p. These systems are solved by a Gauss-Seidel algorithm which requires to solve several systems $\mathcal{H}_g \phi_g = q_g$ for each group g with:

- $\mathcal{H}_g : \phi_g \longrightarrow -div\big(D_g \overrightarrow{\nabla} \phi_g\big) + (\Sigma_{t,g} - \Sigma_s^{g \to g})\phi_g$;
- q_g a function of the right hand side S^p and of the result of the Gauss-Seidel iterations[7].

[7] Let us denote $q_g^{(i)}$ (respectively $\phi_g^{(i)}$) the value of q_g (respectively ϕ_g) at the i^{th} Gauss-Seidel iteration and let us denote S_g^p the restriction of the right hand side S^p to the group g. With these notations, $q_g^{(i)}$ is obtained as following $q_g^{(i)} = S_g^p - \sum_{g'<g} \mathcal{H}^{g' \to g}\, \phi_{g'}^{(i)} - \sum_{g'>g} \mathcal{H}^{g' \to g}\, \phi_{g'}^{(i-1)}$.

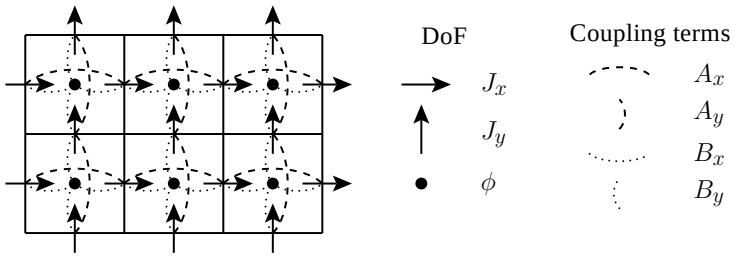


Figure 1: Mesh, degrees of freedom (DoF) and coupling terms with the RT0 finite element for a 2D case.

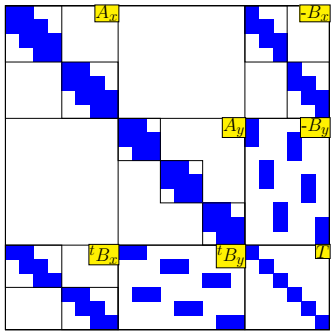


Figure 2: Matrix of the diffusion system.

2.2. Spatial discretization

For the spatial discretization, the same approach as in [7] is used: for each power iteration, for each energy group, the following diffusion problem written with the mixed formulation (which implies the introduction of a new variable $\vec{J_g}$ called current) is solved:

Problem 2. *Find ϕ_g and $\vec{J}_g$, respectively the flux and the current, such that:*

$$\begin{cases} \text{div}\,(\vec{J_g}) + \Sigma_g \phi_g = q_g & \text{in } \Omega \\ \dfrac{\vec{J}_g}{D_g} + \vec{\nabla}\phi_g = \vec{0} & \text{in } \Omega \\ \phi = 0 & \text{on } \partial\Omega \end{cases} \tag{3}$$

where $\Sigma_g = \Sigma_{t,g} - \Sigma_s^{g\to g}$.

Remark : In the following of the paper, g will be ignored to simplify the notations.

A Cartesian mesh[8] is used to discretize equation (3). For each mesh cell K_m, the value D_m (resp. Σ_m) of the function D (resp. Σ) is computed. The Raviart-Thomas finite element (RTk) defines the approximation spaces V_h and W_h [11, 12]. The discrete mixed dual variational formulation of (3) is:

Problem 3. *Find $(\phi, \vec{J}) \in V_h \times W_h$ such that:*

$$\begin{cases} a_h(\overrightarrow{J}, \overrightarrow{w}) - b_h(\phi, \overrightarrow{w}) = 0 & \forall \overrightarrow{w} \in W_h \\ b_h(v, \overrightarrow{J}) + t_h(\phi, v) = q_h(v) & \forall v \in V_h \end{cases} \quad \text{with} \quad \left| \begin{array}{l} a_h(\overrightarrow{J}, \overrightarrow{w}) = \displaystyle\sum_m \frac{1}{D_m} \int_{K_m} \overrightarrow{J}.\overrightarrow{w}\, d\Omega \\ b_h(\phi, \overrightarrow{w}) = \displaystyle\int_\Omega \phi\ div(\overrightarrow{w})\ d\Omega \\ t_h(\phi, v) = \displaystyle\sum_m \Sigma_m \int_{K_m} \phi\ v\ d\Omega \\ q_h(v) = \displaystyle\int_\Omega q\ v\ d\Omega \end{array} \right. .$$

The Raviart-Thomas finite element properties lead to a very sparse algebraic system (see Figure 2). The following unknowns and coupling terms (see Figure 1) are defined as follows:

- the vector ϕ represents the degrees of freedom of the flux;

[8] In [9, 10], the method has been adapted for hexagonal meshes.

Algorithm 1: Global algorithm

Inverse power iteration algorithm
- `/* Inversion of` $\mathcal{H}$ `by Gauss-Seidel method */`
- **Block Gauss-Seidel algorithm**
  - **foreach** *group* g **do**
    - `/* Inversion of` $\mathcal{H}_g$ `by Gauss-Seidel method */`
    - **Block Gauss-Seidel algorithm**
      - **foreach** *space direction* d **do**
        - Compute J_d^g by backward-forward elimination;
    - Compute ϕ_g ;

- the vector J_d represents the degrees of freedom of the current in direction d ($d = x, y$ or z);
- the vector Q is associated to the linear form q_h and contains the source terms;
- the matrix $diag(A_x, A_y, A_z)$ is associated to the bilinear form a_h. It is a block diagonal matrix, as there are no coupling term between the directions. A_d is a symmetric positive definite band matrix (tridiagonal with RT0 approximation and pentadiagonal with RT1 approximation);
- the positive-definite diagonal matrix T is associated to the bilinear form t_h;
- the sparse rectangular matrix $\begin{pmatrix} B_x \\ B_y \\ B_z \end{pmatrix}$ is associated to the bilinear form b_h. B_d contains the coupling terms between J_d and ϕ.

So for a 2D case with a the spatial domain Ω, a linear system of the following form has to be solved :

$$\begin{pmatrix} A_x & & -B_x \\ & A_y & -B_y \\ {}^tB_x & {}^tB_y & T \end{pmatrix} \begin{pmatrix} J_x \\ J_y \\ \phi \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ Q \end{pmatrix}. \tag{4}$$

The flux $\phi = T^{-1}(Q - {}^tB_x J_x - {}^tB_y J_y)$ is eliminated to solve the system (4). Hence the current unknowns are obtained by a Block Gauss Seidel algorithm applied to the following symmetric positive definite system:

$$\begin{pmatrix} W_x & B_x T^{-1} {}^tB_y \\ B_y T^{-1} {}^tB_x & W_y \end{pmatrix} \begin{pmatrix} J_x \\ J_y \end{pmatrix} = \begin{pmatrix} B_x T^{-1} Q \\ B_y T^{-1} Q \end{pmatrix} \quad \text{with} \quad \left| \begin{array}{l} \forall d \in \{x, y\} \\ W_d = A_d + B_d T^{-1} {}^tB_d \end{array} \right. .$$

The matrices W_d and A_d have the same pattern. The resolution of the linear systems involving W_d is based on a Cholesky factorization. The matrices $B_d T^{-1} {}^tB_{d'}$ are not stored in memory as their products are computed by three successive sparse products.

2.3. Overall algorithm

The overall algorithm (see **Algorithm 1**) is made of three nested convergence loops (one for the power inverse algorithm and two for the inner Gauss-Seidel algorithm). In our applications, we obtain for a given accuracy the best performance in terms of CPU time by fixing the number of iterations of all Gauss-Seidel loops to one [6]. In the general SPn case, one level of iterations is added to solve the coupled diffusion systems by block Gauss-Seidel loops over the harmonics.

3. Description of the domain decomposition algorithm

In this section we first describe the previous attempt to parallelize this specific algorithm based on an alternating direction method. Then, we describe the common part between our previous work and the present work. Indeed, the two works are based on a common numerical methodology using Lagrange multipliers. The two works differs by the parallelization scheme, which has a crucial influence on numerical stability and parallel efficiency. Hence to conclude this section, we present the characteristics of our new approach.

3.1. State of the art

To get the first parallelization of the algorithm presented in section 2, the author of [13] used the independence of each current line of J_d (see Figure 1). From an algebraic point of view, the band matrices W_d are block diagonal; each block can be treated in parallel. Due to the structure of the coupling matrix B_y (see Figure 2), which implies a global vector reordering, a large amount of communication is required. As the performance of the computers and the size of the problem to be solved have changed since 1999, it is difficult to estimate the speed-up we could obtain now. However the communication speed increases more slowly than computational speed, and as the method was already limited by the amount of communications, we could not expect to reach good performance with such a parallelization scheme.

To avoid these global communications, domain decomposition methods are well suited. As the problem to solve is an eigenvalue problem, the first attempts to use domain decomposition were based on modal synthesis: in each subdomain several eigen modes have to be computed for each energy group. These modes are used to build the approximation space and its basis. The dimension of the approximation space is small. The resulting small eigenvalue problem is solved sequentially. In [14], an approach with overlapping subdomains called CMS[9] was developed to avoid the difficulties due to the choice of interface modes [13]. To reduce the number of modes to compute, the method was improved with FCMS[10]. Only the first mode is computed; the following modes are replaced by the products of the first mode by sinusoidal functions. Due to the overlap, 8 processors are required to be as efficient as the sequential Minos solver. The algorithm is not very scalable as:

- the size of the overlap increases with the number of subdomains;
- the global eigenvalue problem is treated sequentially and its size increases with the number of subdomains.

So the next attempts were to solve only the linear systems $\mathcal{H}\psi^{p+1} = S^p$ by a domain decomposition strategy. A success was obtained with a domain decomposition method based on a Schwarz multiplicative algorithm with Robin transfer conditions [14]. Some pretreatment steps were not yet parallelized. So for a 3D PWR, and by using between 2 and 18 processors, an efficiency between 60% and 80 % were reached. The author expected an efficiency near 100% once the pre-treatment part parallelized. A first drawback of this method is the introduction of a small overlap of one cell between subdomains due to the discontinuity of the flux on the interface[11]. The second and main drawback of this algorithm comes from the need for a good estimation of the parameter involved in the Robin transfer conditions. *Trial and error* is the only known method to choose this parameter in a good way.

3.2. Preliminary study

In order to overcome these difficulties, we studied a non-overlapping domain decomposition method based on Lagrange multipliers [5, 1] such that

[9] Component Mode Synthesis.
[10] Factorized Component Mode Synthesis.
[11] Coming from the use of the Finite Element RTk.

- it does not introduce new parameters;
- the local solvers are identical to the global sequential solver: in terms of code *reusability* this is a key point because we do not need to develop and maintain a new solver;
- it allows to deal with non-matching grids.

To get the most generic implementation, we chose an approach with an application of the multi-group solver for each subdomain (see **Algorithm 3**). The numerical results obtained with cross sections coming from a real industrial application show an increase in the number of the outer power iterations. Thus, the parallel efficiency was clearly not sufficient. Therefore, we propose an other location of the domain decomposition loop to benefit from the characteristics of the alternating direction method. Before we introduce our the new approach, the common numerical characteristics between the both approaches are described.

The space Ω is cut into two non-overlapping subdomains[12] Ω_1 and Ω_2 such that $\Omega = \Omega_1 \cup \Omega_2$ and $\Gamma = \partial\Omega_1 \cap \partial\Omega_2 = \Omega_1 \cap \Omega_2$. The interface Γ is directed by the normal vector $\overrightarrow{n}$. To obtain the continuous multidomain formulation (**Problem 4**), the equations (3) are rewritten in each subdomain with a new boundary condition on the interface Γ. $(\phi_1, \vec{J}_1)$ (respectively $(\phi_2, \vec{J}_2)$) represents the flux and the current unknowns in the subdomain Ω_1 (respectively Ω_2). ϕ_Γ represents the flux on the interface. To get the equivalence with **Problem 2**, the boundary condition (6) is added to ensure the continuity of the normal component of the current $\overrightarrow{J}$:

Problem 4. *Find* $(\phi_1, \vec{J}_1)$ *and* $(\phi_2, \vec{J}_2)$ *such that:*

$$\left\{\begin{array}{rcll} \mathrm{div}\,(\vec{J}_1) + \Sigma\phi_1 &=& q & \textit{in } \Omega_1 \\ \dfrac{\vec{J}_1}{D} + \vec{\nabla}\phi_1 &=& \vec{0} & \textit{in } \Omega_1 \\ \phi_1 &=& \phi_\Gamma & \textit{on } \Gamma \\ \phi_1 &=& 0 & \textit{on } \partial\Omega_1 \cap \partial\Omega \end{array}\right. \quad \left\{\begin{array}{rcll} \mathrm{div}\,(\vec{J}_2) + \Sigma\phi_2 &=& q & \textit{in } \Omega_2 \\ \dfrac{\vec{J}_2}{D} + \vec{\nabla}\phi_2 &=& \vec{0} & \textit{in } \Omega_2 \\ \phi_2 &=& \phi_\Gamma & \textit{on } \Gamma \\ \phi_2 &=& 0 & \textit{on } \partial\Omega_2 \cap \partial\Omega \end{array}\right. \tag{5}$$

$$\vec{J}_1 \cdot \vec{n} = -\vec{J}_2 \cdot \vec{n} \quad \textit{on } \Gamma. \tag{6}$$

The new variable ϕ_Γ in (5) leads commonly [15, 16, 17] to the introduction of Lagrange multipliers Λ. So the variational formulation of **Problem 4** is **Problem 5**. In [18, 19], this formulation is called mixed-dual hybrid method, but it was not used to obtain an efficient parallel domain decomposition algorithm:

Problem 5. *Find* $(\phi_1, \vec{J}_1) \in L^2(\Omega_1) \times H(\mathrm{div}\,, \Omega_1)$, $(\phi_2, \vec{J}_2) \in L^2(\Omega_2) \times H(\mathrm{div}\,, \Omega_2)$ *and* $\Lambda \in H^{1/2}(\Gamma)$ *such that:*

$$\left\{\begin{array}{ll} \displaystyle\int_{\Omega_1} \mathrm{div}\,(\vec{J}_1)\, v_1 \, d\Omega_1 + \int_{\Omega_1} \Sigma_1\phi_1 v_1 \, d\Omega_1 = \int_{\Omega_1} q_1 v_1 \, d\Omega_1 & \forall\, v_1 \in L^2(\Omega_1) \\ \displaystyle\int_{\Omega_1} \frac{\vec{J}_1}{D_1} \cdot \vec{w}_1 \, d\Omega_1 - \int_{\Omega_1} \phi_1 \,\mathrm{div}\,(\vec{w}_1)\, d\Omega_1 + \int_\Gamma \Lambda \vec{w}_1 \cdot \vec{n} \, d\Gamma = 0 & \forall\, \vec{w}_1 \in H(\mathrm{div}\,, \Omega_1) \\ \displaystyle\int_{\Omega_2} \mathrm{div}\,(\vec{J}_2)\, v_2 \, d\Omega_2 + \int_{\Omega_2} \Sigma_2\phi_2 v_2 \, d\Omega_2 = \int_{\Omega_2} q_2 v_2 \, d\Omega_2 & \forall\, v_2 \in L^2(\Omega_2) \\ \displaystyle\int_{\Omega_2} \frac{\vec{J}_2}{D_2} \cdot \vec{w}_2 \, d\Omega_2 - \int_{\Omega_2} \phi_2 \,\mathrm{div}\,(\vec{w}_2)\, d\Omega_2 - \int_\Gamma \Lambda \vec{w}_2 \cdot \vec{n} \, d\Gamma = 0 & \forall\, \vec{w}_2 \in H(\mathrm{div}\,, \Omega_2) \\ \displaystyle\int_\Gamma \left(\vec{J}_1 - \vec{J}_2\right) \cdot \vec{n}\, \mu \, d\Gamma = 0 & \forall\, \mu \in H^{1/2}(\Gamma) \end{array}\right. .$$

[12] For the sake of simplicity, the presentation of the method is limited to two subdomains. The generalization to more subdomains is pretty obvious.

Each subdomain is now discretized on a Cartesian mesh. On each cell K_m^i, D_i (resp. Σ_i) has the constant value D_i^m (resp. Σ_i^m). **Problem 5** becomes after discretization:

Problem 6. *Find* $(\phi_1, \vec{J_1}) \in V_h^1 \times W_h^1$, $(\phi_2, \vec{J_2}) \in V_h^2 \times W_h^2$ *and* $\Lambda_h \in V_h^\Gamma$ *such that:*

$$\begin{cases} a_1^h(\overrightarrow{J_1}, \overrightarrow{w_1}) - b_1^h(\phi_1, \overrightarrow{w_1}) = -c_1^h(\Lambda_h, \overrightarrow{w_1}) & \forall \overrightarrow{w_1} \in W_h^1 \\ b_1^h(v_1, \overrightarrow{J_1}) + t_1^h(\phi_1, v_1) = q_1^h(v_1) & \forall v_1 \in V_h^1 \\ \\ a_2^h(\overrightarrow{J_2}, \overrightarrow{w_2}) - b_2^h(\phi_2, \overrightarrow{w_2}) = -c_2^h(\Lambda_h, \overrightarrow{w_2}) & \forall \overrightarrow{w_2} \in W_h^2 \\ b_2^h(v_2, \overrightarrow{J_2}) + t_2^h(\phi_2, v_2) = q_2^h(v_2) & \forall v_2 \in V_h^2 \\ \\ c_1^h(\mu_h, \overrightarrow{J_1}) + c_2^h(\mu_h, \overrightarrow{J_2}) = 0 & \forall \mu_h \in V_h^\Gamma \end{cases}$$

$$\textit{with} \quad \left| \begin{array}{l} a_i^h(\overrightarrow{J_i}, \overrightarrow{w_i}) = \displaystyle\sum_m \frac{1}{D_m^i} \int_{K_m^i} \overrightarrow{J_i}.\overrightarrow{w_i} d\Omega \\ \\ b_i^h(\phi_i, \overrightarrow{w_i}) = \displaystyle\int_{\Omega_i} \phi_i \,\mathrm{div}\, (\overrightarrow{w_i})\, d\Omega \end{array} \right. \quad \textit{and} \quad \left| \begin{array}{l} t_i^h(\phi_i, v_i) = \displaystyle\sum_m \Sigma_m^i \int_{K_m^i} \phi_i \; v_i \; d\Omega \\ q_i^h(v_i) = \displaystyle\int_{\Omega_i} q_i \; v_i \; d\Omega \\ c_i^h(\Lambda_h, \overrightarrow{w_i}) = (-1)^{i+1} \displaystyle\int_\Gamma \Lambda_h \vec{w}_i \cdot \vec{n} \; d\Gamma \end{array} \right. .$$

The approximation spaces V_h^i and W_h^i come from the finite element discretization scheme. In case of matching grids (and with the same finite element order used in each subdomain), V_h^Γ is chosen as the trace of W_h^i. This choice provides the equivalence between **Problem 3** and **Problem 6**. In case of non-matching grids, a Mortar technique [16, 20] is used: V_h^Γ is the trace of W_h^1 or W_h^2; the chosen subdomain is called the master subdomain. In our case, the master subdomain is the subdomain with the finest mesh and the highest finite element order. The numerical applications presented in this paper concern only matching grid cases.

In each subdomain Ω_i, the same matrices and vectors are defined as with the monodomain approach:

- the vector J_i^d contains the current degrees of freedom of Ω_i in direction d;
- the vector ϕ_i contains the flux degrees of freedom of Ω_i;
- the vector Q_i contains the source terms of Ω_i;
- the matrix $diag(A_i^x, A_i^y, A_i^z)$ is associated to the bilinear form a_i^h;
- the positive definite diagonal matrix T_i is associated to the bilinear form t_i^h;
- the sparse rectangular matrix $\begin{pmatrix} B_i^x \\ B_i^y \\ B_i^z \end{pmatrix}$ is associated to the bilinear form b_i^h.

New vectors and matrices are introduced to couple the subdomains:

- the vector Λ_d contains the degrees of freedom of the Lagrange multipliers situated on Γ_d (the part of interface directed by the normal vector $\overrightarrow{n_d}$);
- the sparse rectangular matrix $diag(C_{\Lambda\to i}^x, C_{\Lambda\to i}^y, C_{\Lambda\to i}^z)$ is associated to the bilinear form c_i^h. $C_{\Lambda\to i}^d$ contains the coupling term between Λ_d and J_i^d. As the coupling terms between the degrees of freedom inside Ω_i and Λ_d are null, these matrices are sparse.

So, **Problem 6**, in a 2D case, leads to the following algebraic system:

$$\left(\begin{array}{ccc|ccc|cc} A_1^x & & -B_1^x & & & & C^x_{\Lambda\to 1} & \\ & A_1^y & -B_1^y & & & & & C^y_{\Lambda\to 1} \\ {}^tB_1^x & {}^tB_1^y & T_1 & & & & & \\ \hline & & & A_2^x & & -B_2^x & C^x_{\Lambda\to 2} & \\ & & & & A_2^y & -B_2^y & & C^y_{\Lambda\to 2} \\ & & & {}^tB_2^x & {}^tB_2^y & T_2 & & \\ \hline {}^tC^x_{\Lambda\to 1} & & & {}^tC^x_{\Lambda\to 2} & & & & \\ & {}^tC^y_{\Lambda\to 1} & & & {}^tC^y_{\Lambda\to 2} & & & \end{array}\right)\left(\begin{array}{c} J_1^x \\ J_1^y \\ \phi_1 \\ \hline J_2^x \\ J_2^y \\ \phi_2 \\ \hline \Lambda_x \\ \Lambda_y \end{array}\right)=\left(\begin{array}{c} 0 \\ 0 \\ Q_1 \\ \hline 0 \\ 0 \\ Q_2 \\ \hline 0 \\ 0 \end{array}\right). \tag{7}$$

3.3. The new approach

In this section, we adopt a different strategy from [1], where the authors tried to get a more generic approach by using a multigroup solver in each subdomain.

In the same way as the sequential solver, we benefit from the specificity of Raviart Thomas finite element to perform the flux elimination in each subdomain:

$$\phi_i = T_i^{-1}(Q_i - {}^tB_i^x J_i^x - {}^tB_i^y J_i^y). \tag{8}$$

Hence, the currents J_i^d satisfy the following equations:

$$\left(A_i^d + B_i^d T_i^{-1\,t} B_i^d\right) J_i^d + \left(B_i^d T_i^{-1\,t} B_i^{d'}\right) J_i^{d'} + C^d_{\Lambda\to i}\Lambda_d = B_i^d T_i^{-1} Q_i.$$

By letting $W_i^d = A_i^d + B_i^d T_i^{-1\,t} B_i^d$ and $W_i^{d,d'} = B_i^d T_i^{-1\,t} B_i^{d'}$, we obtain the following system:

$$\left(\begin{array}{cc|cc|cc} W_1^x & W_1^{x,y} & & & C^x_{\Lambda\to 1} & \\ W_1^{y,x} & W_1^y & & & & C^y_{\Lambda\to 1} \\ \hline & & W_2^x & W_2^{x,y} & C^x_{\Lambda\to 2} & \\ & & W_2^{y,x} & W_2^y & & C^y_{\Lambda\to 2} \\ \hline {}^tC^x_{\Lambda\to 1} & & {}^tC^x_{\Lambda\to 2} & & & \\ & {}^tC^y_{\Lambda\to 1} & & {}^tC^y_{\Lambda\to 2} & & \end{array}\right)\left(\begin{array}{c} J_1^x \\ J_1^y \\ \hline J_2^x \\ J_2^y \\ \hline \Lambda_x \\ \Lambda_y \end{array}\right)=\left(\begin{array}{c} B_1^x T_1^{-1} Q_1 \\ B_1^y T_1^{-1} Q_1 \\ \hline B_2^x T_2^{-1} Q_2 \\ B_2^y T_2^{-1} Q_2 \\ \hline 0 \\ 0 \end{array}\right). \tag{9}$$

Through a matrix reordering, (9) becomes:

$$\left(\begin{array}{ccc|ccc} W_1^x & & C^x_{\Lambda\to 1} & W_1^{x,y} & & \\ & W_2^x & C^x_{\Lambda\to 2} & & W_2^{x,y} & \\ {}^tC^x_{\Lambda\to 1} & {}^tC^x_{\Lambda\to 2} & & & & \\ \hline W_1^{y,x} & & & W_1^y & & C^y_{\Lambda\to 1} \\ & W_2^{y,x} & & & W_2^y & C^y_{\Lambda\to 2} \\ & & & {}^tC^y_{\Lambda\to 1} & {}^tC^y_{\Lambda\to 2} & \end{array}\right)\left(\begin{array}{c} J_1^x \\ J_2^x \\ \Lambda_x \\ \hline J_1^y \\ J_2^y \\ \Lambda_y \end{array}\right)=\left(\begin{array}{c} B_1^x T_1^{-1} Q_1 \\ B_2^x T_2^{-1} Q_2 \\ 0 \\ \hline B_1^y T_1^{-1} Q_1 \\ B_2^y T_2^{-1} Q_2 \\ 0 \end{array}\right).$$

This 2×2 block matrix[13] is solved by a block Gauss-Seidel algorithm. As the extra-diagonal blocks are block diagonal, the multiplication by these blocks can easily be computed in parallel. The remaining issue is to solve the following linear system (the right hand side comes from the Gauss-Seidel algorithm) for each direction d :

$$\left(\begin{array}{cc|c} W_1^d & & C^d_{\Lambda\to 1} \\ & W_2^d & C^d_{\Lambda\to 2} \\ \hline {}^tC^d_{\Lambda\to 1} & {}^tC^d_{\Lambda\to 2} & \end{array}\right)\left(\begin{array}{c} J_1^d \\ J_2^d \\ \hline \Lambda_d \end{array}\right)=\left(\begin{array}{c} F_1^d \\ F_2^d \\ \hline 0 \end{array}\right). \tag{10}$$

[13] In 3D, it is a 3×3 block matrix.

We introduce the notations[14]:

$$\hat{W}^d = \begin{pmatrix} W_1^d & \\ & W_2^d \end{pmatrix} \quad \hat{C}^d = \begin{pmatrix} C^d_{\Lambda\to 1} \\ C^d_{\Lambda\to 2} \end{pmatrix} \quad \hat{J}^d = \begin{pmatrix} J_1^d \\ J_2^d \end{pmatrix} \quad \hat{F}^d = \begin{pmatrix} F_1^d \\ F_2^d \end{pmatrix}.$$

In each direction, we have to solve the saddle point system:

$$\begin{pmatrix} \hat{W}^d & \hat{C}^d \\ {}^t\hat{C}^d & \end{pmatrix} \begin{pmatrix} \hat{J}^d \\ \Lambda_d \end{pmatrix} = \begin{pmatrix} \hat{F}^d \\ 0 \end{pmatrix}. \tag{11}$$

In this way, the new domain decomposition strategy leads to a mono-dimensional domain decomposition for each matrix W^d.
Remark : In the following of the paper, the indices d are ignored.

Λ is the solution of the interface system

$$\left({}^t\hat{C}\hat{W}^{-1}\hat{C}\right)\Lambda = {}^t\hat{C}\hat{W}^{-1}\hat{F} \tag{12}$$

which is solved with a Preconditioned Conjugate Gradient algorithm. Then $\hat{J}$ is computed by $\hat{J} = \hat{W}^{-1}\left(\hat{F} - \hat{C}\Lambda\right)$. **Algorithm 2** describes these two operations. This iterative algorithm is initialized by Λ_0 which is the value of the Lagrange multipliers Λ at the previous iteration of the outer algorithm.

Let us now introduce the multidomain version to present the preconditioner. With a partition in N subdomains in one direction (illustrated by Figure 3 in the RT0 case), the matrices and vectors $\hat{W}$, $\hat{J}$, $\hat{F}$, $\hat{C}$ and Λ are rewritten as following:

$$\hat{W} = \begin{pmatrix} W_1 & & & \\ & W_2 & & \\ & & \ddots & \\ & & & W_N \end{pmatrix} \quad \hat{J} = \begin{pmatrix} J_1 \\ J_2 \\ \vdots \\ J_N \end{pmatrix} \quad \hat{F} = \begin{pmatrix} F_1 \\ F_2 \\ \vdots \\ F_N \end{pmatrix};$$

$$\hat{C} = \begin{pmatrix} C_{\Lambda_1^2\to 1} & & \\ C_{\Lambda_1^2\to 2} & \ddots & \\ & \ddots & C_{\Lambda^N_{N-1}\to N-1} \\ & & C_{\Lambda^N_{N-1}\to N} \end{pmatrix} \quad \Lambda = \begin{pmatrix} \Lambda_1^2 \\ \Lambda_2^3 \\ \vdots \\ \Lambda_{N-1}^N \end{pmatrix}.$$

- Λ_i^j denotes the degrees of freedom of the Lagrange multipliers of the interface Γ_i^j between the subdomains Ω_i and Ω_j;
- $C_{\Lambda_i^j\to i}$ is the matrix that contains the coupling terms between the Lagrange multipliers Λ_i^j and the subdomain Ω_i.

With $S^k_{i\to j} = {}^t\hat{C}_{\Lambda_k^j\to j}\hat{W}_j^{-1}\hat{C}_{\Lambda_i^j\to j}$ the matrix involved in (12) called S is:

$$S = {}^t\hat{C}\hat{W}^{-1}\hat{C} = \begin{pmatrix} \left(S^1_{1\to 2} + S^2_{2\to 1}\right) & S^1_{3\to 2} & & \\ S^3_{1\to 2} & \ddots & \ddots & \\ & \ddots & \ddots & S^{N-2}_{N\to N-1} \\ & & S^N_{N-2\to N-1} & \left(S^{N-1}_{N-1\to N} + S^N_{N\to N-1}\right) \end{pmatrix}.$$

[14]The upper-script symbol $\hat{}$ distinguishes the assembled matrix W on the domain Ω and the matrix $\hat{W}$ which contains all local W_i matrices.

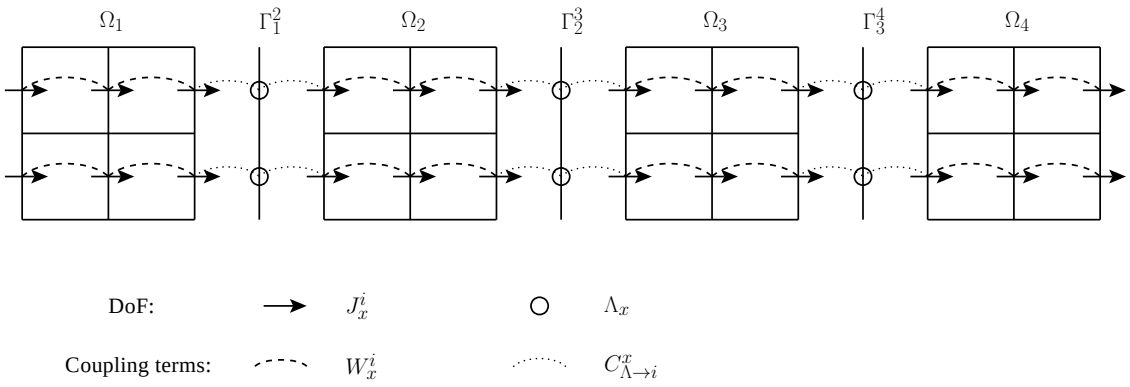


Figure 3: Ω is partitioned in $N = 4$ subdomains into the direction x. As we are interested in solving by domain decomposition the problem in the direction x after flux elimination, the flux ϕ_i, the current J_i^y, the coupling terms B_i^d and A_i^y are ignored.

We choose as preconditioner the block diagonal preconditioner:

$$P^{-1} = \begin{pmatrix} \left(S_{1\to 2}^1 + S_{2\to 1}^2\right) & & \\ & \ddots & \\ & & \left(S_{N-1\to N}^{N-1} + S_{N\to N-1}^N\right) \end{pmatrix}.$$

Usually this preconditioner is expensive in terms of computational time or memory consumption, because all the diagonal blocks are dense. By using the Raviart-Thomas properties, an effective implementation of this preconditioner is provided in section 4.2.

3.4. Summary

Let us reconsider the algorithm of the previous approach [5, 1]. Compared to the sequential monodomain algorithm, a new loop is added to solve the saddle point system induced by domain decomposition (see **Algorithm 3**). This loop was put between the power algorithm and the loop over the energy groups. As the inner loops (iterative algorithms in each subdomains) are fixed point algorithms with only one iteration, we also used a fixed point method to solve the saddle point system (Uzawa-MR algorithm was chosen). The previous approach was more generic than the new one, but the increase in the number of outer iterations for real industrial cases was not satisfying.

In the new approach, as in the previous one, one loop is added to **Algorithm 4** to get convergence of the iterative algorithm used to solve the interface systems. In our new approach the domain decomposition loop is under the alternate direction computations and use the solver in each spatial direction. Hence we can benefit from a direct solver in each subdomain. In the following, we consider a fixed number i of iterations: the algorithm is called `PCGi`. In the same way as the sequential monodomain solver, the iteration loop of the outer algorithm is used to get convergence. The new approach enables some new implementation optimizations which are described in the following section.

The both approachs are compared in the manuscrit [**?**] of the PhD which is at the origin of this article.

4. Optimized implementation on matching grids

The optimizations presented in this section are based on the sparse pattern of the matrices $S_{i\to j}^k$. A call to the local matrix W_j^{-1} is usually required to compute the multiplication by the matrix $S_{i\to j}^k$. So the computation complexity is linear with the number of current degrees of freedom.

Algorithm 2: PCG

In : $\hat{F}$
Inout: Λ Initialized to Λ_0
Out : $\hat{J}$

1 $\hat{J} = \hat{W}^{-1}\left(\hat{F} - \hat{C}\Lambda\right)$;
2 $g = \left({}^t\hat{C}\hat{J}\right)$;
3 $z = Pg$;
$w = z$; $dotGZ = < g, z >$;
while *! Convergence* **do**
6 $S_w = ({}^t\hat{C}\hat{W}^{-1}\hat{C})w$;
$\rho = -\dfrac{dotGZ}{< w, S_w >}$;
8 $\Lambda = \Lambda + \rho w$;
9 $g = g + \rho S_w$;
10 $z = Pg$;
$dotGZold = dotGZ$; $dotGZ = < g, z >$;
12 $w = z + \dfrac{dotGZ}{dotGZold} w$;
end
14 $\hat{J} = \hat{W}^{-1}\left(\hat{F} - \hat{C}\Lambda\right)$

With matching grids, all current lines are independent (see Figure 3), so the matrices $S^k_{i\to j}$ are diagonal[15]. By storing these matrices, the complexity of a product is linear with the number of degrees of freedom of the Lagrange multipliers between the subdomain Ω_i and Ω_j. To build these matrices, their products by $\mathbb{1}$, the interface vector filled of 1, are computed by performing a forward/backward substitution. These matrices are computed once at the first iteration of the power inverse iteration algorithm. So the cost of this computation is reduced with the number of power inverse iterations.

The method detailed in this section could be generalized to the non-matching grid case; in the case where the finest mesh is included in the coarse mesh with a ratio r, the matrices $S^k_{i\to j}$ are band with $2r^{Dim} - 1$ bandwidth.

4.1. Product by the interface matrix

To compute S_ω (line 6 of **Algorithm 2**), the product of S by the interface vector $\omega = {}^t(\omega^2_1, \omega^3_2, \ldots, \omega^N_{N-1})$, it is necessary to compute four terms for each interface. For an interface Γ^r_l between the left subdomain Ω_l and the right subdomain Ω_r, the component $(S_\omega)^r_l$ is decomposed in two parts coming from the contribution of the two subdomains in order to reduce the number of communications. If l' denotes the subdomain $l-1$ and r' denotes the subdomain $r+1$, we have:

$$\begin{aligned}(S_\omega)^r_l &= S^l_{l\to l'}\omega^{l'}_l + \left(S^l_{l\to r} + S^r_{r\to l}\right)\omega^r_l + S^r_{r'\to r}\omega^{r'}_r \\ &= \left(S^l_{l\to l'}\omega^{l'}_l + S^l_{l\to r}\omega^r_l\right) + \left(S^r_{r\to l}\omega^r_l + S^r_{r\to r'}\omega^{r'}_r\right).\end{aligned} \tag{13}$$

As the matrices $S^k_{i\to j}$ are stored, the call to the local matrix W_i^{-1} is avoided and the computational cost is linear with the number of interface degrees of freedom.

[15] With the previous approach there were dense.

Algorithm 3: Global algorithm with domain decomposition : previous approach

```
Inverse power iteration algorithm
    /* Inversion of saddle point problem by Uzawa-MR method                    */
    Uzawa-MR algorithm
        /* Inversion of H by Gauss-Seidel method                               */
        Block Gauss-Seidel algorithm
            foreach group g do
                /* Inversion of H_g by Gauss-Seidel method                     */
                Block Gauss-Seidel algorithm
                    foreach space direction do
                        foreach subdomains do
                            Perform backward-forward elimination;
                Compute φ_g ;
```

Algorithm 4: Global algorithm with domain decomposition : new approach

```
Inverse power iteration algorithm
    /* Inversion of H by Gauss-Seidel method                                   */
    Block Gauss-Seidel algorithm
        foreach group g do
            /* Inversion of H_g by Gauss-Seidel method                         */
            Block Gauss-Seidel algorithm
                foreach space direction do
                    Preconditioned Conjugate Gradient algorithm
                        foreach subdomains do
                            Perform backward-forward elimination;
            Compute φ_g ;
```

4.2. The preconditioner

As P^{-1}, and so P, are diagonal matrices, the block diagonal preconditioner is the diagonal preconditioner. Only one interface vector is necessary to store P and the application of P (line 3 and 10 of **Algorithm 2**) is almost free as it consists in term by term products. Then for each interface Γ_i^j, the two contributions coming from $S_{i\to j}^i$ and $S_{j\to i}^j$ are added to obtain P^{-1}. So the computational cost is low since the complexity is linear with the number of interface degrees of freedom. It is not necessary to introduce a more sophisticated preconditioner. We use only `PCG1` as numerical experiments have shown that it is sufficient in practice. We can imagine to use a higher number of iterations to by-pass possible convergence difficulties[16]. Indeed an additional iteration is not very costly, as the product by the interface matrices S is almost free in terms of computational time.

4.3. Solution rebuilding

The expensive call to the local matrix W_i^{-1} usually implied by the interface matrix (line 6 of **Algorithm 2**) is already avoided. As a very small number of iterations is required for practical cases, we are interested in reducing the computational cost of the solution rebuilding (line 14 of **Algorithm 2**). The operation is decomposed in two parts, one already computed (line 1 of

[16]Even on industrial cases with heterogeneous data, we did not encounter this kind of difficulties.

Algorithm 2) and one that takes into account the modifications of the Lagrange multiplier Λ (line 8 of **Algorithm 2**):

$$\hat{J} = \hat{W}^{-1}\left(\hat{F} - \hat{C}\Lambda_0\right) + \hat{W}^{-1}\hat{C}\Delta \quad \text{with } \Delta = \Lambda_0 - \Lambda.$$

The second term $\tilde{J} = \hat{W}^{-1}\hat{C}(\Delta)$ can be computed efficiently as it requires to compute in each subdomain Ω_i (with a left subdomain l and a right subdomain r) the following term:

$$\tilde{J}_i = (W_i^{-1}C_{\Lambda_i^l \to i})\Delta_i^l + (W_i^{-1}C_{\Lambda_i^r \to i})\Delta_i^r. \tag{14}$$

For a current line indiced by c the restriction of a vector V to this current line is noted $V_{|c}$ and for an interface vector ω this restriction is a scalar denoted $\omega[c]$. Thanks to the current lines independence, (14) is equivalent to:

$$\forall c, \quad \tilde{J}_{i|c} = \Delta_i^l[c].(W_i^{-1}C_{\Lambda_i^l \to i}\mathbb{1})_{|c} + \Delta_i^r[c].(W_i^{-1}C_{\Lambda_i^r \to i}\mathbb{1})_{|c}. \tag{15}$$

So the computation of $\hat{J}$ (line 14 of **Algorithm 2**) done by forward and backward substitutions using Cholesky factorization in each subdomain is replaced by a linear combination of three vectors in each subdomain. The overhead cost of computing the vector $(W_i^{-1}C_{\Lambda_i^j \to i}\mathbb{1})$ is zero as these vectors are required to compute the matrix $S_{i\to j}^i$.

4.4. The parallel communication scheme

The spatial domain Ω is divided into $N_x \times N_y$ subdomains for a 2D case. Each process stores the data of its subdomain for each energy group and for each harmonic in order to avoid communication of complete spatial fields. The interface vectors are duplicated: each adjacent subdomain has its own storage, such that the two versions of a vector are numerically exactly equivalent. The computations on the interface vectors (lines 8, 9 and 12 of **Algorithm 2**) are duplicated[17] in order to minimize the communications.

The communication scheme is based on two communication types:

- *point to point communications*, which are necessary to exchange data across the interfaces. It is only necessary to transfer interface vectors. As the preconditioner P is also duplicated, the only point to point communications come from the product by ${}^t\hat{C}$ (line 2 of **Algorithm 2**) and from the product by the interface matrix (line 6). **Algorithm 5** describes the first operation using asynchronous communications. The second operation uses the same communication scheme[18].

- *global communications*, which are necessary to compute dot products. Two families of communicators are involved :

  - The communicator `CommWorld`[19] contains all the $N_x \times N_y$ processors. This communicator is used to compute the dot products of the outer algorithms, through a reduction operation such as `MPI_Allreduce`. The outer Gauss-Seidel algorithm (over energy groups) may require dot products to evaluate the stopping criteria. The inverse power algorithm requires also dot products to compute the eigenvalue k_{eff}, to perform Chebyshev acceleration and to evaluate the stopping criteria.
  - In each process, a communicator is created for each space direction. For a 2D case, these communicators are called `CommX` and `CommY` (see Figure 4). `CommX` (respectively `CommY`) contains N_x (respectively N_y) MPI processes. Indeed, as all current lines are independent, the Conjugate Gradient algorithm can be applied to a current line or

[17] Performed on two subdomains.
[18] The line 8 is modified to take into account the terms coming from (13).
[19] Usually called `MPI_COMM_WORLD`.

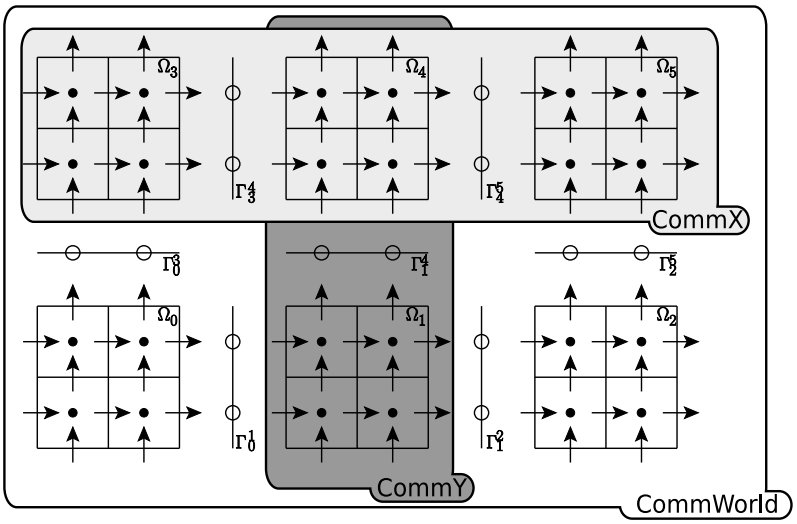


Figure 4: Ω_4 communicators: Ω is decomposed in six subdomains (three in the direction x and two in the direction y). The different communicators involved by the subdomain Ω_4, are `CommX`, `CommY` and `CommWorld`.

to a group of lines. In order to minimize the size of communications and the size of the communicators, the Conjugate Gradient algorithm is applied to the group of current lines induced by the Cartesian domain partition. So the communicator `CommX` (respectively `CommY`) regroups these lines in the direction x (respectively y).

If all current lines were regrouped, the communicator (usually `CommWorld`) would be larger . If the Conjugate Gradient algorithm is applied independently on each current line, the `MPI_Allreduce` is applied on a vector. Its size is the number of current lines in the communicator `CommX` (or `CommY`).

Algorithm 5: Product by ${}^t\hat{C}$

executed by: process i.
input: J_i the current vector of Ω_i.
output: ω_i^l and ω_i^r such as $\omega = {}^t\hat{C}J$. with l, r the left/right interfaces.
for $j \in \{l, r\}$ **do**
  Throw asynchronous receive in vector $\omega Recv_i^j$ from process j;
end
for $j \in \{l, r\}$ **do**
  $\omega_i^j = {}^t\hat{C}_{\Lambda_j^i \to i} J_i$;
  $\omega Send_i^j = \omega_i^j$;
  Throw asynchronous send of $\omega Send_i^j$ to process j;
end
for $j \in \{l, r\}$ **do**
  `/* The order of loop iteration depends on the order of reception */`
  Wait the reception of $\omega Recv_i^j$;
  $\omega_i^j += \omega Recv_i^j$;
end
Wait all data sent ;

5. Numerical validation

In this section, numerical results are provided:

- for the academic 3D IAEA-BenchMark [21] in section 5.1;
- for cross-sections provided by a real industrial application in section 5.2. With this test case, we want to evaluate the behavior of the method with non-homogeneous data (each pin in an assembly has its own section).

The tests have been performed in double precision on a cluster with 208 nodes (each one consists in two Intel Xeon processors, 3.40GHz, 2MB Cache and with 4 GB PC3200 DDR2). The used MPI implementation is MPICH 1.2.7 with an Infiniband (openib-2.0.5) network. For time measurement, during batch submissions, one node is assigned to each subdomain to avoid concurrent memory accesses. The current implementation of the algorithm is adapted to a distributed-memory context, with one subdomain per node. Hence the performance analysis is performed up to 169 subdomains (see Figure 5(d), Figure 5(e), Figure 6(d), Figure 6(e) and Figure 7). To get numerical results such as the number of iterations of the power inverse algorithm (see Figure 5(a) and Figure 6(a)), or a comparison to monodomain solver (see Figure 5(b), Figure 5(c), Figure 6(b) and Figure 6(c)), several subdomains per nodes can be used. So the numerical analysis is performed until 961 subdomains.

Between two and eight subdomains, the domain is partitioned in the direction x. For more subdomains the partition is balanced as much as possible between the direction x and the direction y. The domain is not partitioned into the direction z as:

- there are few cells (38 or 40) in the direction z;
- the number of subdomains should divide the number of cells to keep a good load balance. Results with tridimensional partitions of 36 and 144 subdomains are provided for the second test case (section 5.2) which uses 40 cells in direction z.
- a coupling with monodimensional thermo-hydraulic module will be the next step. The flow is considered as axial (into the direction z), so no radial communication (into the direction x and y) will be required by this module.

For the two cases, the solution obtained by the multi-domain solver is compared to the reference solution $(\psi^{ref}, k_{eff}^{ref})$ obtained by the sequential solver set up with a large number (2000) of outer iterations. The solution is also compared to $(\psi^{mono}, k_{eff}^{mono})$ the solution obtained by the sequential solver with the same stopping criteria as the multi-domain solver. This stopping criteria for the inverse power algorithm is:

$$\frac{\|S^p - S^{p-1}\|}{\|S^p\|} < \epsilon_S \quad \text{and} \quad \frac{|\lambda^p - \lambda^{p-1}|}{|\lambda^p|} < \epsilon_\lambda, \tag{16}$$

where S^p and λ^p are respectively the source of fission $\mathcal{F}\psi^p$ and the estimated eigenvalue k_{eff} at iteration p. To perform a comparison of eigenvectors, each vector is normalized with the euclidean norm.

5.1. 3D IAEA BenchMark

To validate our method, the two groups homogeneous IAEA-3D BenchMark [21] is first considered. The stopping criteria is set to $\epsilon_\lambda = \epsilon_S = 10^{-6}$. For a SP1-RT0 pin-by-pin computation, the mesh consists of $289 \times 289 \times 38$ cells; this leads to:

- 3 173 798 degrees of freedom for the flux for each energy group;
- 9 626 879 degrees of freedom for the current for each energy group;
- 25 601 354 degrees of freedom for the overall system.

5.1.1. Numerical behavior

For the majority of domain partitions, the power algorithm satisfies the stopping criteria with a reasonable number of iterations like the sequential mono-domain solver did (see Figure 5(a)); for partitions between 1 and 121 subdomains, with `PCG1` and `PCG2`, the outer algorithm satisfies the stopping criteria with 78 iterations (like the sequential solver). Between 144 and 625 subdomains a small increase ($< 12\%$) is observed with `PCG1`, and with `PCG2` the number of iterations remains constant. For more than 625 subdomains the algorithm does not converge with a reasonable number of iterations (150) with `PCG1`. By using `PCG2` the convergence properties are improved: the power inverse algorithm converges with less than 88 iterations up to 961 subdomains.

For a given stopping criteria, the accuracy of the multi-domain solution is the same as the accuracy of the mono-domain solver[20] (see Figure 5(b) and 5(c)). As expected, the use of `PCG2` provides a solution closer to the mono-domain solver but not closer to the reference solver.

5.1.2. Computational efficiency

Once the accuracy verified, the performance of the method can be analyzed. With the implementation described in the previous section, the computational overhead compared to the sequential solver is small. At each inner iteration, this overhead consists of a linear combination of three current vectors (15) and interface vector operations. So we can expect a parallel efficiency near 100%. Figure 5(d) shows this efficiency. The algorithm `PCG1` leads to excellent results with an efficiency higher than the theoretical 100% as it can benefit from cache effects. The algorithm `PCG2` is less efficient, as there is the same number of outer iterations. As the implementation of the product by the interface matrix is efficient, the difference remains small. So, all performance analysis are performed with `PCG1` algorithm, since the number of subdomains (available number of nodes) is limited to 169. Figure 5(e) shows the time repartition between:

- the forward-backward substitutions involved by the local matrices $\hat{W}^{-1}$. In the first node the local matrix is noted W_0^{-1};
- the computations involving the matrices B_i^d. These contain the flux rebuilding (8) and the right-hand side $\hat{F}$ which comes from the alternating direction algorithm based on a Gauss-Seidel algorithm;
- the communications.

The super-linearity comes from the computation of the multiplications by the matrices B_d and ${}^tB_d$, necessary to compute the right-hand side $\hat{F}$ and the flux ϕ_i. Indeed, the ordering of the current unknowns J_d, which is very efficient for the matrices W_d, requires non contiguous memory accesses during operations with B_d and ${}^tB_d$. With more than 32 nodes, the subdomains are sufficiently small to benefit from cache effects. The results are very good but we can not expect to maintain this efficiency with many more than 169 nodes. Above 169 nodes, the communications can become a significant bottleneck.

5.2. Real industrial case

We consider cross-sections provided by a real industrial application with two energy groups. The stopping criteria is set up with $\epsilon_\lambda = \epsilon_S = 10^{-5}$. SP1 equations with a pin-by-pin discretization ($289 \times 289 \times 40$ cells) and RT0 finite element lead to:

- 3 340 840 degrees of freedom for the flux for each energy group;
- 10 129 161 degrees of freedom for the current for each energy group;
- 26 940 002 degrees of freedom for the overall system.

[20] The difference between this accuracy and the stopping criteria is acceptable for our application.

partition	outer iterations	time (in s)	$\lvert k_{eff} - k_{eff}^{ref} \rvert$	$\lVert \psi - \psi^{ref} \rVert_2$
(6, 6, 1)	88	$6,680$	$2,64.10^{-7}$	$1,63.10^{-3}$
(4, 9, 1)	88	$6,749$	$2,66.10^{-7}$	$1,63.10^{-3}$
(9, 4, 1)	88	$6,731$	$2,64.10^{-7}$	$1,62.10^{-3}$
(36, 1, 1)	830	$70,17$	$4,42.10^{-7}$	$2,38.10^{-3}$
(1, 36, 1)	826	$68,89$	$4,59.10^{-7}$	$2,42.10^{-3}$
(6, 3, 2)	88	$6,912$	$2,63.10^{-7}$	$1,63.10^{-3}$
(3, 6, 2)	88	$6,884$	$2,63.10^{-7}$	$1,63.10^{-3}$
(9, 1, 4)	88	$7,144$	$2,60.10^{-7}$	$1,62.10^{-3}$
(1, 9, 4)	88	$7,271$	$2,61.10^{-7}$	$1,62.10^{-3}$
(3, 3, 4)	88	$7,090$	$2,63.10^{-7}$	$1,63.10^{-3}$

partition	outer iterations	time (in s)	$\lvert k_{eff} - k_{eff}^{ref} \rvert$	$\lVert \psi - \psi^{ref} \rVert_2$
(12, 12, 1)	88	$1,548$	$2,92.10^{-7}$	$1,54.10^{-3}$
(8, 9, 2)	88	$1,588$	$2,60.10^{-7}$	$1,62.10^{-3}$
(9, 8, 2)	88	$1,606$	$2,59.10^{-7}$	$1,61.10^{-3}$
(6, 6, 4)	88	$1,695$	$2,63.10^{-7}$	$1,63.10^{-3}$

Table 1: Influence of the domain partition.

These cross sections are identical to those in [1] where the algorithm strongly suffers from convergence difficulties.

In Figure 6, the same behavior as for the IAEA BenchMark is observed. It is a great improvement compared to the results obtained in [1] (an efficiency higher than 100% is measured now where an efficiency near 5% was obtained before). The fact that the number of outer iterations is equal, or almost equal, to the sequential case proves that it is not useful to study more complex preconditioners. Indeed, even with an optimal preconditioner, we can not expect to further reduce the outer iterations number compared to a direct solver.

Compared to the IAEA BenchMark, one difference can be noted: with `PCG1` the error of the eigenvalue k_{eff} increases slowly with the number of subdomains (see Figure 6(b)). Actually this issue has no effect for our application where an accuracy of the order 10^{-4} is sufficient. With a large number of subdomains, the number of power inverse iterations is almost equal to number of iterations of the sequential solver (see Figure 6(a)).

Table 1 shows the performance with different partitions for 36 and 144 subdomains. With monodimensional partitions convergence is difficult: more than 800 power iterations are required to satisfy the stopping criteria. So bidimensional and tridimensional partitioning have to be considered. The experiment shows that it is not efficient to partitioned the domain in the direction z. Indeed, with a tridimensional partition the size of the interface vectors and the number of communications increase. The basic partitions $(6, 6, 1)$ and $(12, 12, 1)$ give the best results for this experiment. The number of outer iterations and the execution time is not necessary equivalent for symmetric partitions. Indeed the Gauss-Seidel algorithm of the alternating direction method introduces an asymmetry.

The method works perfectly well with the SP3 equations (2 harmonics) and with RT1 finite element. The efficiency[21] is plotted in Figure 7 for SP1/RT0 (26 940 002 dof), SP1/RT1 (214 666 888 dof) and SP3/RT0 (53 880 004 dof). The results with SP3/RT1 (429 333 776 dof) are not presented as we are not able to run the sequential reference even on a special node with 16 Go of memory.

With the RT1 discretization, the efficiency is less impressive but remains high. The RT0 implementation can benefit from the super linearity of the *reshuffle* operations with more than 32 processors. So, with the RT1 finite element, this gain can be excepted with at least $32 \times 2^3 = 256$ processors. We need to perform experiments on a larger cluster[22] to confirm this explanation.

With SP3 equations a better efficiency is measured. We explain that by the larger number of multiplications by B_d and ${}^tB_d$ matrices which are used to couple the two diffusion systems for each energy group. So the super-linearity effects are more effective.

6. Conclusions

The proposed domain decomposition method in the difficult context of an approximate resolution of the linear system at each inverse power iteration reveals very reliable and efficient on SP1/SP3 pin-by-pin computation coming from the IAEA benchmark and from industrial cases. The method satisfies two criteria: *memory requirement* and *parallel efficiency*. Compared to [1], the parallel efficiency is improved by a factor 20. The method is as least as efficient as the result obtained in [14] and avoids the difficulties resulting from:

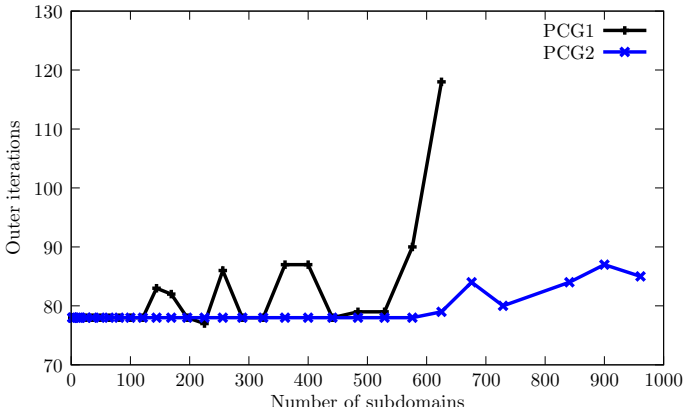
- the choice of the Robin transfer coefficient;
- the introduction of an additional approximation and a small overlap due to the Robin transfer conditions with a mixed dual formulation.

As the numerical behavior of the method is very satisfactory with a large number of subdomains, a next step will be a hybrid implementation (shared and distributed memory). Such an implementation will enable us to use more efficiently a multi-core architecture. This step is needed to get an efficient non-matching grid implementation, since for non-matching grids more flexibility is required to balance the load between the nodes.

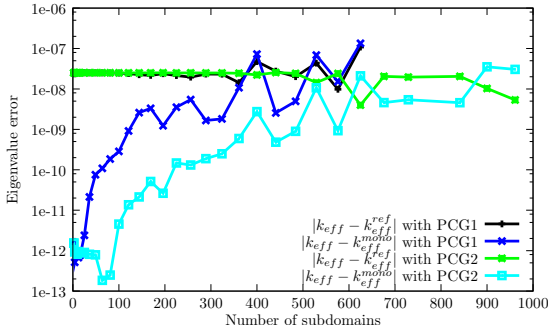
Hardware accelerators such as Graphic Processor Units suffer from insufficient memory size but can be very relevant in terms of computational time. In [22], the authors obtained a speed-up of 30 for the monodomain solver. So, it seems natural to use the domain decomposition algorithm on a GPU cluster in order to take advantage of GPU acceleration without memory limitation.

The next step is the integration of this parallel algorithm in the industrial platform. Hence it will be possible to perform studies with a large number of groups and with pin-by-pin discretizations.

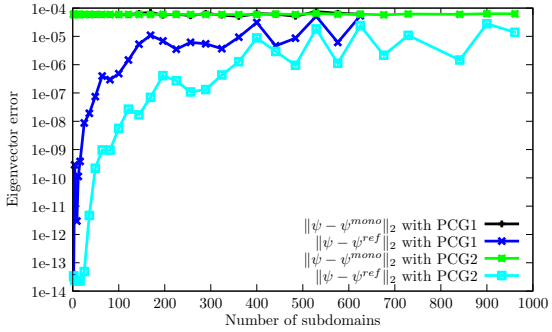
[21] The sequential reference time is obtained on a special node with 16 Go of memory.
[22] or on a cluster with a larger cache size.



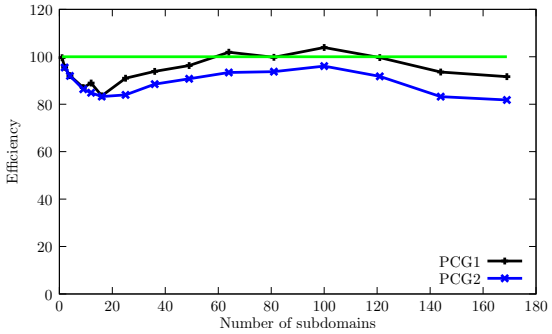
(a) Number of outer iterations



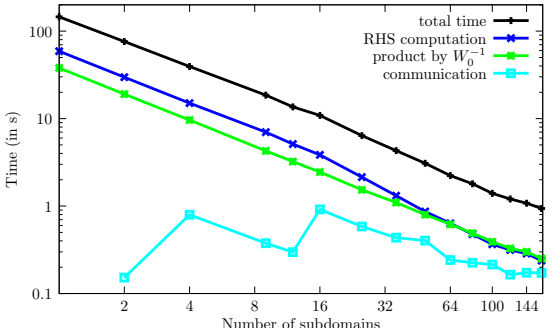
(b) Eigenvalue error



(c) Flux eigenvector error

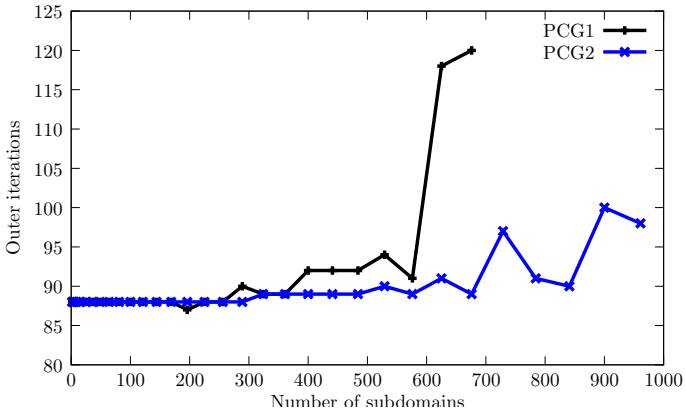


(d) Parallel Efficiency

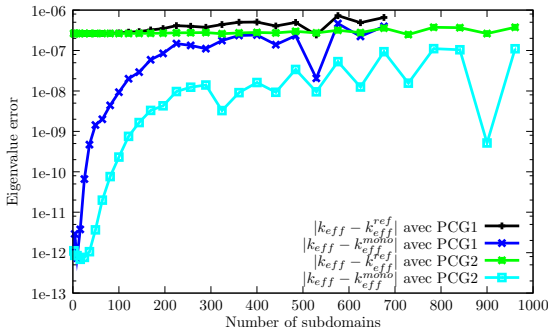


(e) Detailed execution time

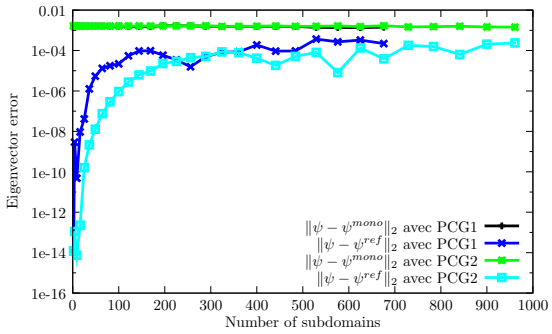
Figure 5: BenchMark IAEA.



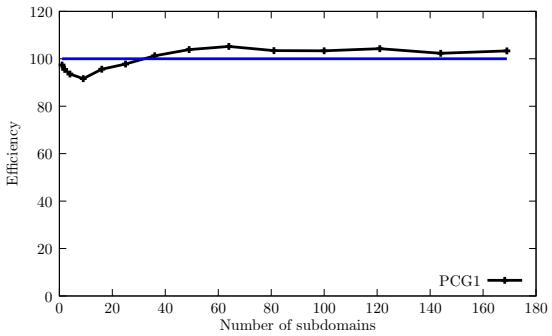
(a) Number of outer iterations



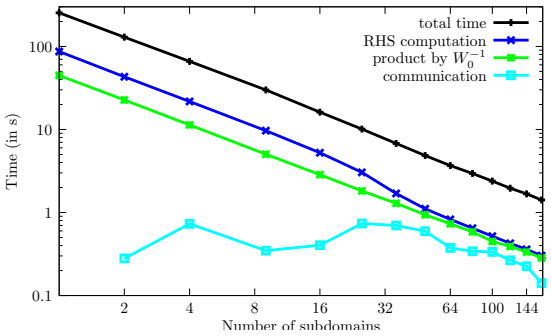
(b) Eigenvalue error



(c) Flux eigenvector error



(d) Parallel Efficiency



(e) Detailed execution time

Figure 6: Real industrial case.

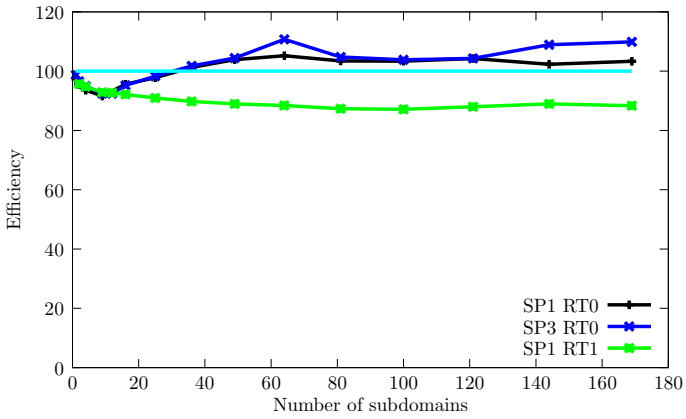


Figure 7: Efficiency with SP1/SP3 equations and RT0/RT1 finite element.

References

[1] B. Lathuilière, M. Barrault, P. Ramet, J. Roman, A non overlapping parallel domain decomposition method applied to the simplified transport equations, in: Proceedings of Mathematics, Computational Methods & Reactor Physics, 2009.
URL `http://www.labri.fr/~ramet/restricted/mc09.pdf.gz`

[2] L. Plagne, A. Ponçot, Generic programming for deterministic neutron transport codes, in: Proceedings of Mathematical and Computation, Supercomputing, Reactor Physics and Nuclear and Biological Applications, Palais des Papes, Avignon, France, 2005.

[3] E. Gelbard, Simplified spherical harmonics equations and their use in shielding problems, Technical Report WAPD-T-1182, Bettis Atomic Power Laboratory (1961).

[4] C. Pomraning, Asymptotic and variational derivation of the simplified Pn equations, Annals of Nuclear Energy 20 (1993) 623–637.

[5] M. Barrault, B. Lathuilière, P. Ramet, J. Roman, A domain decomposition method applied to the simplified transport equations, in: Proceedings IEEE CSE'08, 11th IEEE International Conference on Computational Science and Engineering, São Paulo, SP, Brazil, 2008, pp. 91–97.

[6] J. Lautard, F. Moreau, A fast 3D parallel solver based on the mixed dual finite element approximation, in: Mathematics & Computation and Super Computing in Nuclear Applications, 1993.

[7] A. Baudron, J. Lautard, D. Schneider, Mixed dual methods for neutronic reactor core calculations in the CRONOS system, in: Reactor Physics and Environmental Analysis of Nuclear Systems, 1999.

[8] E. Lewis, W. Miller, Computational Methods of Neutron Transport, Wiley, New York, 1984.

[9] D. Schneider, Éléments finis mixtes duaux pour la résolution numérique de l'équation de la diffusion neutronique en géométrie hexagonale, Ph.D. thesis, Université Paris VI (2001).

[10] A. Hébert, Mixed-dual implementations of the simplified Pn method, Annals of Nuclear Energy 37 (2010) 498–511.

[11] P. Raviart, J. Thomas, A mixed finite element method for second order elliptic problems, Lecture Notes in Mathematics.

[12] J. C. Nédélec, A new family of mixed finite elements in $\mathbb{R}^3$, Numerische Mathematik 50 (1986) 57–81.

[13] K. Pinchedez, Calcul parallèle pour les équations de diffusion et de transport homogènes en neutronique, Ph.D. thesis, Université Paris XI (1999).

[14] P. Guérin, Méthodes de décomposition de domaine pour la formulation mixte duale du problème critique de la diffusion des neutrons, Ph.D. thesis, Université Paris VI (2007).

[15] Swann, On the use of lagrange multipliers in domain decomposition for solving elliptic problems, Mathematics of Computation 60 (201) (1993) 49–78.

[16] C. Bernardi, Y. Maday, A. T. Patera, A New Non Conforming Approach to Domain Decomposition: The Mortar Element Method, in: H. Brezis, J.-L. Lions (Eds.), Collège de France Seminar, Pitman, 1994, this paper appeared as a technical report about five years earlier.

[17] B. Wohlmuth, Discretization methods and iterative solvers based on domain decomposition, Springer, 2001.

[18] F. Coulomb, C. Fedon-Magnaud, Mixed and mixed-hybrid elements for the diffusion equation, Nuclear Science and Engineering 100 (3) (1988) 218–225.

[19] S. Van Criekingen, R. Beauwens, Mixed-hybrid discretization methods for the P1 equations, Applied Numerical Mathematics 57 (2) (2007) 117–130.

[20] F. Ben Belgacem, Discrétisations 3D non conformes pour la méthode de décomposition de domaine des éléments avec joints : analyse mathématique et mise en oeuvre pour le problème de Poisson, Ph.D. thesis, EDF (1993).

[21] B. Micheelsen, 3D IAEA Benchmark Problem, Tech. rep., IAEA (1977).

[22] W. Kirschenmann, L. Plagne, S. Ploix, A. Ponçot, S. Vialle, Massively parallel solving of 3D simplified P_N equations on graphic processing units, in: Proceedings of Mathematics, Computational Methods & Reactor Physics, 2009.